Password Organizer and Generator Menggunakan Algoritma Genetik

Teddy Marcus Zakaria, Daniel Dananjaya
Jurusan S1 Teknik Informatika
Fakultas Teknologi Informasi, Universitas Kristen Maranatha
Jl. Prof. Drg. Surya Sumantri No. 65 Bandung 40164
E-mail: Teddy.mz@maranatha.edu; danan2004@gmail.com

Abstract

Password, are something confidential and only had by some people that relational. As a human being, we sometimes forgot password we had made or changed. Because of that, a helpful device was needed to organize password. Password Organizer and Generator are one of many application that can help to create a password and organize password which inputed by user. Genetic algorithm was used to create password, so the password strength were created became strong and more secure. This application store password in XML form. The data were encrypted with algorithm PBE with MD5 and DES so the data would be more secure or keep confidential.

Keywords : password, organizer, generator password, genetic algorithm, PBE with MD5 and DES, password strength

1. Pendahuluan

Password atau dikenal juga dengan sebutan kata sandi adalah sebuah susunan huruf yang diperlukan seseorang untuk dapat masuk ke dalam sebuah aplikasi atau untuk dapat masuk ke dalam sebuah halaman tersendiri di internet seperti *e-mail*. Kata sandi tersebut mungkin kita lupa. Untuk itu kita harus memasukkan sebuah kata sandi baru yang dapat kita ingat lagi secara lebih baik. Kata sandi biasanya digunakan untuk dapat masuk ke dalam aplikasi yang hanya boleh diketahui oleh orang yang bersangkutan sehingga kata sandi tersebut tidak diperbolehkan diketahui oleh orang lain.

Adanya keterbatasan yang dimiliki manusia, menyebabkan sulitnya bagi kita untuk mengingat hal-hal seperti *password* maupun *username*. Apalagi, jika kita tidak sering menggunakan *username* dan *password* tersebut, biasanya kita harus mendaftar lagi atau mengganti dengan sebuah *password* yang baru. Hal ini sering membuat kita kesal karena harus berulang-ulang melakukan hal yang sama jika membutuhkan untuk masuk ke dalam halaman web terutama disaat kita butuh secara cepat.

2. Tujuan

Aplikasi yang akan dibuat memiliki tujuan untuk memudahkan *user* yang memiliki banyak *password* yang harus diingat. *User* hanya butuh untuk mengingat sebuah

password yang akan digunakan untuk masuk ke dalam aplikasi. Selain itu, aplikasi ini juga dapat menyarankan sebuah *password* baru yang dapat digunakan. Kekuatan dari *password* juga akan diukur dan diberitahukan kepada *user* agar *user* dapat mengganti *password* yang kurang kekuatannya dengan sebuah *password* baru yang lebih kuat.

3. Spesifikasi Produk

Aplikasi ini hanya membagi satu tingkatan saja yaitu *user*. *User* dapat melakukan hal – hal berikut :

- Memasukkan *username*, *password* dan alamat *website*.
  Tiga data tersebut dapat ditambah dan diubah sesuai kebutuhan *user*.
- *Log In*
  User dapat mengaktifkan dan masuk ke dalam aplikasi ini.
- Melihat alamat *website* yang disimpan.
  Data alamat web dapat ditampilkan untuk dapat dilihat oleh *user* halaman web mana saja yang telah di simpan dalam aplikasi.
- Menghapus data.
  User dapat menghapus data untuk halaman *website* yang tidak ingin disimpan.
- Melihat daftar *password* yang dapat digunakan.
  User dapat memilih *password* yang dihasilkan sistem untuk digunakan.

Untuk melihat daftar *password* yang dihasilkan oleh sistem, digunakan algoritma genetik. Sebuah kromosom diasumsikan sebagai sebuah *password*. Proses Pembuatan password akan dilakukan melalui langkah berikut :

1. Pembuatan populasi awal.

Pembuatan populasi awal yang akan ditetapkan sebanyak 200 populasi. Populasi ini dibuat dengan melakukan pengulangan sebanyak yang diperlukan dari 1000 kromosom yang dibuat secara acak. Populasi ini akan terdiri dari kromosom dengan nilai *fitness* yang lebih dari 190. Contoh kode program pembuatan populasi awal dapat dilihat pada gambar populasi awal.

```
private void setSomeBestChromosome(){
    while(bestChromosome.size() < 200){
        createPopulation();
        for (int a=0;a<population.length;a++){
            //System.out.print(" ");
            if(fitness[a]>190){
                someBestChromosome.add(population[a]);
                someBestFitness.add(fitness[a]);
                bestChromosome.add(population[a]);
                bestFitness.add(fitness[a]);
            }
        }
    }
    System.out.println(bestChromosome.size());
    sortBestChromosome();
}
```

Gambar 1. Populasi awal

2. Penghitungan *fitness* dan pengurutan.
   Penghitungan *fitness* telah dilakukan pada saat pembuatan populasi untuk mempersingkat waktu, sehingga pada langkah kedua dilakukan pengurutan. Pengurutan dilakukan dengan menggunakan algoritma *bubble sorting* secara *descending*. Langkah pengurutan ini dilakukan untuk mendapatkan rangking dari kromosom berdasarkan nilai *fitness*nya.
3. Proses mutasi dan perkawinan.
   Proses mutasi dilakukan dengan memilih secara acak selain lima kromosom yang memiliki nilai *fitness* paling tinggi, menduplikasinya kemudian mengubah salah satu gene dengan membalikkan satu nilai bit yang dipilih secara acak.
   Contoh kode program untuk melakukan mutasi dapat dilihat pada gambar mutasi.

```
private Chromosome doMutation() {
    int mutateIndx = 0;
    Chromosome newChromosome = new Chromosome(population[5+rd.nextInt(
    mutateIndx = rd.nextInt(newChromosome.getChromosomeLength());
    newChromosome.mutateIt(mutateIndx);
    return newChromosome;
}
```

Gambar 2. mutasi

Sementara untuk proses perkawinan, dilakukan dengan memilih secara acak dua buah kromosom dari lima kromosom yang memiliki *fitness* paling tinggi. Kemudian dari dua kromosom ini akan dibentuk satu kromosom baru dengan menggabungkan tujuh gene awal yang dimiliki kromosom induk pertama dengan tujuh gene awal kromosom induk kedua.

Contoh kode program untuk melakukan perkawinan dapat dilihat pada gambar perkawinan.

```
private Chromosome CrossOver(Chromosome firstChrom,Chromosome s
    Gene[] newChromosome = new Gene[14];
    for(int i=0;i<7;i++)
        newChromosome[i] = firstChrom.getGeneAtChromosome(i);
    for(int i=0;i<7;i++)
        newChromosome[i+7] = secondChrom.getGeneAtChromosome(i)
    Chromosome tempChromo = new Chromosome(newChromosome);
    return tempChromo;
}
```

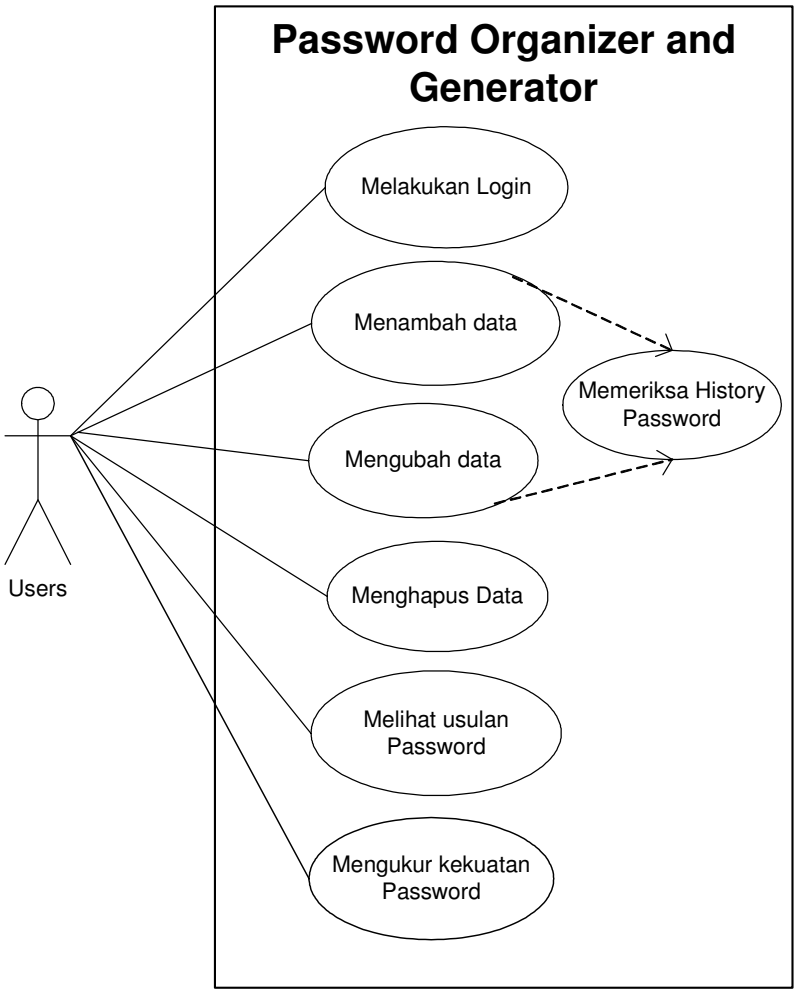
Gambar 3. perkawinan

4. Ulangi mulai dari langkah kedua hingga sebanyak lima puluh kali.

4. Desain Perangkat Lunak

4.1 Usecase

Use case pada aplikasi ini dapat dilihat pada gambar usecase

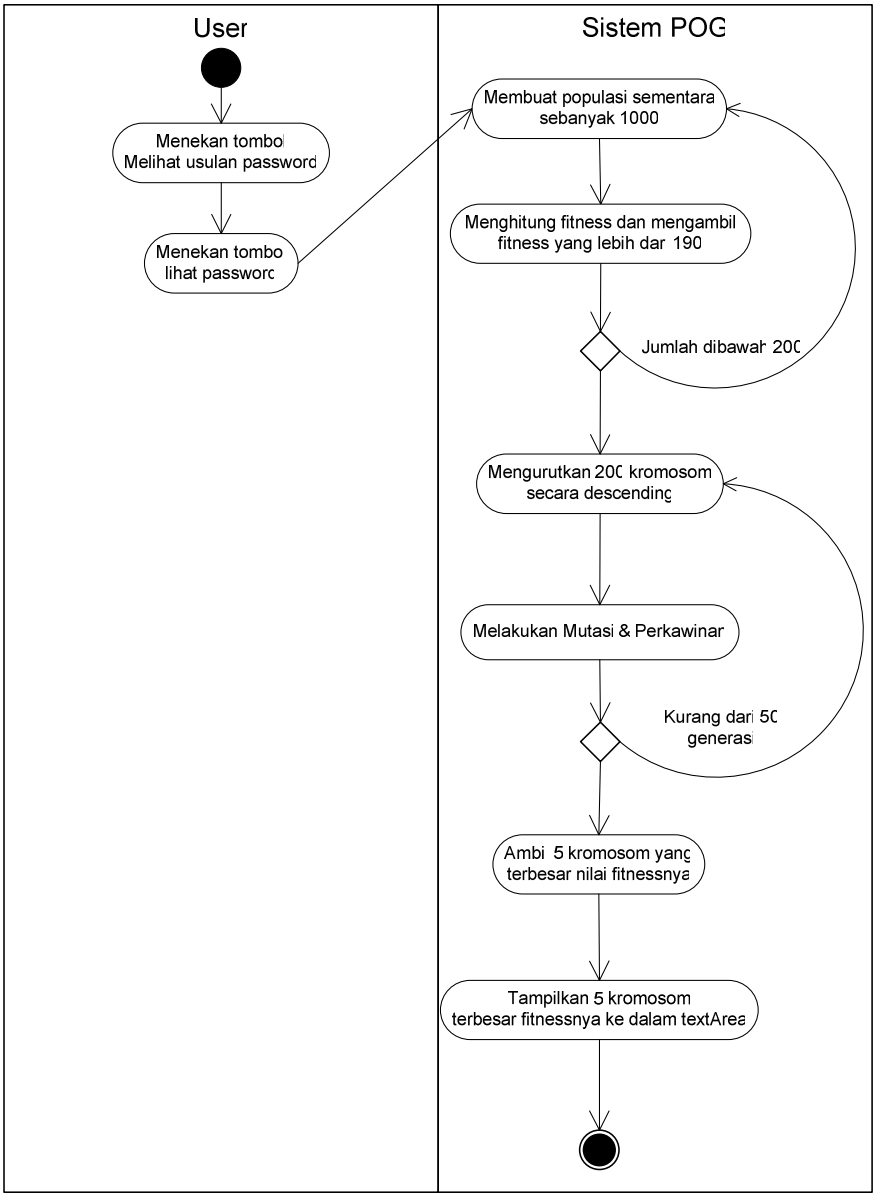


Gambar 4. ***Usecase Diagram***

Gambar *use case diagram,* menjelaskan hal yang dapat dilakukan *user* pada aplikasi *Password organizer and generator* ini. Pada aplikasi ini, seorang *user* terdaftar dapat melakukan *login*, menambahkan data, mengubah data yang telah disimpan dalam aplikasi, melihat usulan *password* yang dihasilkan oleh sistem melalui algoritma genetik, mengukur kekuatan *password* yang dimasukkan oleh *user*.

4.2 *Activity Diagram*

Salah satu aktifitas yang dapat dilakukan oleh user digambarkan pada gambar *activity diagram*.



Gambar 5. ***Activity Diagram***

Database yang digunakan dalam bentuk file *XML*. Desain yang digunakan seperti tampak pada gambar desain database.

```
<AppData>
 <UserData>
  <AppList>
  </AppList>
  <WebList>
   <WebSite>
   </WebSite>
  </WebList>
 </UserData>
</AppData>
```

Gambar 6. Desain D*atabase*

Data untuk masuk ke dalam aplikasi akan disimpan dalam tag <AppList>. Untuk data-data *password* dan *username* serta alamat web akan disimpan dalam tag <WebSite> yang memiliki *parent* tag <WebList>. Selain itu, juga dibuat desain *database* untuk menyimpan seluruh *password* yang pernah digunakan atau disimpan di dalam aplikasi. Desain *database* tersebut dapat dilihat pada gambar desain *password history*.

```
<AppPwd>
 <UserPwd>
  <UserName></UserName>
  <PassList>
   <Password></Password>
  </PassList>
 </UserPwd>
</AppPwd
```

Gambar 7. Desain *Password History*

5. Pengembangan Sistem

5.1 Pembagian *Class* Implementasi

Pembagian *Class* tersebut dibagi menjadi:

1. *Class UserData* berfungsi untuk user yang ingin masuk ke sistem POG ini.
2. *Class WebData* yang berfungsi untuk melakukan manajemen *user*name, password maupun alamat web yang dimasukkan oleh *user*.
3. *Class XMLHandler* digunakan untuk memanagement database yang berisi *user*name, *password* dan alamat web. Sistem kerja *XMLHandler* sendiri mampu membaca dan menulis data kedalam *database* yang berupa *file XML*.
4. *Class AllData* berfungsi untuk menampung keseluruhan data dari web data dan *userdata*

5. *Class Encrypter* yang berfungsi untuk menangani semua kegiatan enkripsi sebelum data ditulis ke dalam *file XML* dan melakukan dekripsi saat data diambil dari *file XML.*

5.2 Keterkaitan Antar *Class*

Sistem POG memiliki beberapa keterkaitan di setiap kelasnya, diantaranya adalah :

- Keterkaitan antara *XMLHandler* dengan *Encrypter* dan *allData* yaitu dimana saat sistem menjalankan *class*, *XMLHandler* mengambil data dari *database*, melakukan proses *decrypt* kemudian memasukkannya ke dalam *class AllData*. Demikian juga yang terjadi ketika akan menuliskan data ke dalam *file XML*, class *XMLHandler* akan menjalankan proses enkripsi melalui *Class Encrypt*. Kemudian menuliskan hasil enkripsi ke dalam *file xml*.
- Keterkaitan antara *AllData* dengan *UserData* dan *WebData* yaitu menampung seluruh data dari *UserData* dan *WebData* di dalam sebuah format yang mudah digunakan di dalam penulisan ke *file XML* maupun pembacaan untuk digunakan oleh sistem.

6. Testing dan Evaluasi Sistem

6.1 *Test Case*

- **Modul *Add data*.** Pada modul ini telah dilakukan tes dengan menambahkan sebuah data baru.
- **Modul *Password Suggestion*.** Pada modul ini telah dilakukan tes dengan membuat sebuah objek baru dari kelas *Genetic* yang menerima dua buah parameter yang digunakan sebagai acuan untuk melakukan mutasi dan perkawinan yang diharapkan telah memuat 1000 buah *password* sebagai awalan.
- **Modul *XMLHandler***. Pada modul ini, telah dilakukan tes pembuatan *file XML* sebagai *database* yang digunakan sistem.

6.2 Ulasan Evaluasi

Secara keseluruhan, aplikasi POG ini sudah cukup memadai. Hal ini dapat dilihat dari aplikasi yang sudah dilengkapi dengan *error handling,* salah satunya adalah ditampilkannya pesan *error* jika ada kesalahan dalam pemasukkan data. Selain itu, tampilan di dalam aplikasi POG disusun secara teratur untuk setiap modul yang ada:

- Menu – menu dari setiap modul ditampilkan pada bagian bawah, dan menu yang dipilih akan ditampilkan pada bagian atas menu dengan menggunakan *panel*, sehingga *user* lebih mudah memakainya.

7. Kesimpulan

Hasil evaluasi dengan *black box* dan *test case* ditarik sebuah kesimpulan yang menjadi masukan-masukan bagi aplikasi *Password Organizer and Generator* ini antara lain :

- Dapat membantu *user* untuk mempermudah dalam mengingat *password* karena hanya perlu mengingat sebuah *master password*.
- Mengurangi tingkat penggantian *password* yang terjadi akibat sering melupakan *password* yang digunakan.
- Mampu untuk memberikan saran *password* yang kuat sehingga menyulitkan seseorang yang ingin melakukan penerobosan.

8. Saran

Selain menarik kesimpulan dari hasil evaluasi didapatkan saran-saran yang dapat mengembangkan aplikasi *Password Organizer and Generator* dimasa yang akan datang. Saran-saran tersebut antara lain adalah :

- *Database* file diubah menjadi sebuah *file XML* yang terenkripsi nama tagnya untuk menambahkan keamanannya.
- Desain antarmuka dapat dikembangkan untuk dapat lebih memudahkan *user*.
- Pengembangan untuk melakukan *login* secara langsung ke dalam sebuah website dengan menekan tombol *hotKeys*.
- Apabila data hilang, dapat melakukan *recovery* melalui log yang dibuat.

Daftar Pustaka

[Cho03] Chonoles, Michael Jesse, James A. Schardt (2003). UML2 FOR DUMMIES. New York Wiley Publishing Inc.

[Ano09A] INTRODUCTION TO XML. http://www.w3schools.com/XML/xml_whatis.asp (26 June 2009)

[Low05] Lowe,Doug, Murach, Joel (2005). MURACH'S BEGINNING JAVA 2 JDK 5. Mike Murach & Asscociate, Inc.

[Mar05] Mark Watson,PRACTICAL ARTIFICIAL INTELEGENCE PROGRAMMING IN JAVA. Version 1.02(18 November 2005)

[Mit02] Mitchell, Melanie (2002). AN INTRODUCTION TO GENETIC ALGORITHMS. New-Delhi: Prentice-Hall of India.

[Phi09] Phiras. Password Strength Meter (A Jquery Plugin). Retrieved May, 28, 2009, from http://phiras.wordpress.com/2007/04/08/password-strength-meter-a-jquery-plugin/

[Tot09] Totheriver.com (2009). Xml and Java – Parsing XML using Tava Tutorial. Retrieved May, 31, 2009, from http://www.java-samples.com/showtutorial.php?tutorialid=152

[Ano09B] -, http://www.passwordmeter.com/ (29 Mei 2009)